|  |
| --- |
| *ООО Эко-Е* |
| Протокол обмена данными ECO-E |
| *Детальное описание реализации протокола обмена данными с устройствами производства ООО Экосфера. Внутренняя документация.* |
| Автор: *Громов Всеволод Владимирович (gromov.vsevolod@yandex.ru)* |
| Ревизия: *19* |

Оглавление

[1. Общее описание. 2](#_Toc287858107)

[2. Описание пакета. 3](#_Toc287858108)

[2.1. Frame type (1 byte). 3](#_Toc287858109)

[2.2. Device address (1 byte). 3](#_Toc287858110)

[2.3. Command (2 byte). 3](#_Toc287858111)

[2.4. Data length (2 byte). 4](#_Toc287858112)

[2.5. Packet number (2 byte). 4](#_Toc287858113)

[2.6. Reserved (2 byte). 4](#_Toc287858114)

[2.7. CRC (header) (2 byte). 4](#_Toc287858115)

[2.8. Data. 4](#_Toc287858116)

[2.9. CRC (data) (2 byte). 4](#_Toc287858117)

[3. Описание команд и ошибок. 5](#_Toc287858118)

[3.1. Пинг. 5](#_Toc287858119)

[3.2. Смена адреса клиента. 5](#_Toc287858120)

[3.3. Установка новой скорости обмена. 5](#_Toc287858121)

[3.4. Запрос на выполнение удаленного вызова (Remote Procedure Call AKA RPC) 5](#_Toc287858122)

[4. Коды ошибок и их описание. 6](#_Toc287858123)

[5. Фрагмент заголовочного файла с номерами команд. 7](#_Toc287858124)

# Общее описание.

Протокол является запрос-ответным, т.е. полудуплексным. Но возможна работа в потоковом варианте в полнодуплексном режиме. Для этого в клиенте имеются циклические входной буфер и обычный выходной.

Инициатором обмена данными между клиентом (далее – «клиент») и управляющим устройством (ПК, КПК и т.п., далее - «мастер»), всегда является «мастер». Обмен данными происходит посредством сообщений (пакетов). мастер посылает пакет-запрос, клиент отвечает либо пакетом-ответом, либо сообщением об ошибке.

Пакеты имеют переменную длину, но не могут быть меньше 12 байт (заголовок пакета должен быть всегда). Исходящий от мастера - не может превышать 512 байт. Исходящие от клиента – не более 512 байт. Однако, при работе в потоковом варианте возможен сбой в работе буфера и при меньших размерах входящих пакетов, если клиент не успевает обрабатывать входящие сообщения. В полудуплексном варианте работы – такой сбой исключен.

Пакет должен содержать четное число байт.

Все контрольные суммы имеют размер 2 байта и рассчитываются согласно общепринятому стандарту (см. [RFC 1071](http://www.faqs.org/rfcs/rfc1071.html)).

# Описание пакета.

Обмен происходит с помощью пакетов. Все пакеты (как запросные, так и ответные) имеют одинаковые поля. Пакет имеет вид:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Номер поля | 1 | | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|  | Frame type | Device address | | Command | Data length | Packet number | Header CRC | Data | Data CRC |
|  | *Static header* | | | | | | | *Data* | |

Первые 7 полей образуют статический заголовок размером 12 байт. Этот заголовок является обязательным элементом. Последние 2 поля образуют поле для передачи данных – это необязательный элемент, размер которого может варьироваться.

## Frame type (1 byte).

Тип пакета – это число (1 байт), которое определяет направление пакета. Сообщения, отсылаемые мастером, маркируются числом 0x7B. От клиента к мастеру – 0x7A. В случае, если клиенты образуют элементарную сеть, то исходящий от мастера пакет принимается всеми клиентами, но отвечает мастеру только тот клиент, чей адрес был указан в этом пакете. Остальные клиенты штатно принимают исходящий от мастера пакет без каких-либо действий. При исходящем пакете от клиента (в случае элементарной сети), этот пакет также полностью принимают как мастер, так и другие клиенты, но они также игнорируют данный пакет.

## Device address (1 byte).

Адрес клиента – это число (1 байт), которое соответствует адресу клиента, который абсолютно необходим при работе в элементарных сетях. Опять же – пакеты с адресом, отличным от клиентского, отбрасываются.

## Command (2 byte).

Каждое сообщение содержит код команды (2 байта). Это число, определяет действия, которые необходимо выполнить клиенту по требованию мастера. Клиент, отвечая на команду с кодом “X”, отсылает мастеру ответный пакет, помещая в это поле ответный код команды “X+0x8000”, – если команда обработана верно. Если ошибка, - то указывает в этом поле код ошибки. Соответственно общее пространство кодов команд и ошибок имеет следующие разделы:

* 0x0000…0x7EFF – команды от мастера к клиенту;
* 0x8000…0xFEFF – ответные команды клиента;
* 0xFF00…0xFFFF – коды ошибок;

## Data length (2 byte).

Данное поле содержит общую длину (в байтах) поля передачи данных (поля 8 и 9!). В случае, если данные передаваться вместе с этим пакетом не будут, данное поле должно быть равно 0.

## Packet number (2 byte).

В этом поле мастер оставляет метку – исходящий номер пакета. Клиент при ответе просто копирует этот номер в такое же поле своего ответного пакета. Данное поле необходимо для реализации потоковой передачи данных в асинхронном режиме (полнодуплексном).

## Reserved (2 byte).

Зарезервировано для возможного использования в будущем.

## CRC (header) (2 byte).

В этом поле указывает контрольная сумма заголовка (2 байта).

## Data.

В этом поле идут собственно данные. Максимальный размер данных может быть таким, чтобы общий размер пакета не превышал указанных в разделе 1 размеров. Также, данное поле должно содержать четное число байт. В случае, если необходимо передать нечетное число байт, данные расширяются добавлением в конец байта 0x00 (со стороны МАСТЕРА!).

## CRC (data) (2 byte).

В этом поле указывает контрольная сумма для поля данных (6) (2 байта). Клиент полностью игнорирует входящ(ий)(ие) пакет(ы) от мастера в случаях:

* неверный CRC заголовка;
* неверное направление пакета;
* неверный адрес клиента;
* пакет слишком длинный;
* переполнение приемного буфера (при асинхронной работе);

В остальных случаях – клиент отвечает либо сообщением об ошибке, либо ответным пакетом.

# Описание команд и ошибок.

На данный момент реализованы 3 базовые команды.

## Пинг.

* Код команды = 0x0000. Дополнительных данных нет.
* Ответный код = 0x8000, без дополнительных данных.

Команда предназначена для проверки наличия канала связи между клиентом и управляющей программой.

## Смена адреса клиента.

* Код команды = 0x0001, дополнительные данные (размер поля данных) – 4 байта.
  + 0 **UINT16 -** новый номер клиента (0-255);
  + 1 **UINT16 -** CRC (данных);
* Ответный код = 0x8001, без дополнительных данных.

Ответ на эту команду формируется со старым адресом клиента, после чего происходит смена адреса клиента и следующая команда от мастера уже обрабатывается с новым адресом клиента.

## Установка новой скорости обмена.

* Код команды = 0x0003, дополнительные данные = 6 байт.
  + 0 **UINT32 -** требуемая скорость обмена
  + 1 **UINT16** - CRC;
* Ответный код: 0x8003, без дополнительной информации.

Ответ на эту команду приходит на старой скорости обмена, после чего происходит переключение на новую скорость и следующая команда от мастера должна быть уже на новой скорости. Если запрашиваемая скорость обмена не поддерживается на клиентской стороне, возвращается код ошибки.

## Запрос на выполнение удаленного вызова (Remote Procedure Call AKA RPC)

* Код команды = 0x0020, размер дополнительных данных варьируется в зависимости от передаваемых параметров, но не меньше 6 байт.
  + 0 **UINT16** – идентификатор RPC;
  + 1 **UINT16** – сигнатура вызова RPC;
  + 2 **UINT16** – CRC;
* Ответный код: 0х8020, данные состоят из идентификатора вызванной процедуры, статуса вызова, опционального стека возвращаемого значения, CRC.

Для ознакомления с деталями реализации подсистемы RPC, см. документ «[Описание подсистемы RPC](Описание%20подсистемы%20RPC.docx)»

# Коды ошибок и их описание.

* 0xFF00 – размер поля данных указанный в заголовке не совпадает с реальным размером поля данных принятого пакета;
* 0xFF01 – неверное выравнивание поля данных (нечетный размер);
* 0xFF02 – ошибка CRC поля данных;
* 0xFF03 – принят пакет с неизвестным кодом команды;
* 0xFF04 – размер поля данных превышает допустимый предел;
* 0xFF05 – размер поля данных не совпадает с ожидаемым для этой команды;
* 0xFF06 – запрашиваемая скорость не поддерживается на стороне клиента;
* 0xFF07 – попытка установить новый адрес клиента, который выходит за допустимые пределы;
* 0xFF08 – устройство не готово к ответу на команду.

# Фрагмент заголовочного файла с номерами команд.

// ECO-E proprietary protocol header.

// idea: Sergei Arestov

// implementation: Vsevolod Gromov

//

// frame types

#define EKOSF\_FT\_MASTER ((BYTE)0x7B)

#define EKOSF\_FT\_SLAVE ((BYTE)0x7A)

// command & error code bases

//

#define EKOSF\_CMD\_REQUEST\_BASE ((WORD)0x0000)

#define EKOSF\_CMD\_RESPONSE\_BASE ((WORD)0x8000)

#define EKOSF\_ERROR\_BASE ((WORD)0xFF00)

// basic commands & error codes

//

// commands...

#define EKOSF\_CMD\_PING (EKOSF\_CMD\_REQUEST\_BASE) // "are you there" ping request

#define EKOSF\_CMD\_ADDR\_SET (EKOSF\_CMD\_REQUEST\_BASE+1) // set new device address

#define EKOSF\_CMD\_RATE\_SET (EKOSF\_CMD\_REQUEST\_BASE+3) // request new data rate. rate is desired bps value, DWORD

#define EKOSF\_CMD\_RPC\_EXEC (EKOSF\_CMD\_REQUEST\_BASE+32) // executes remote procedure call on the responder's side

// data contains WORD == rpc id,

// followed by packed rpc stack

// respond either with:

// EKOSF\_ERR\_CMD\_UNKNOWN (really bad, we cannot do much with such device)

// EKOSF\_ERR\_NOTREADY device is busy executing another procedure, etc.

// EKOSF\_CMD\_RPC\_EXEC + EKOSF\_CMD\_RESPONSE\_BASE, followed by response stack

// service macros

//

#define EKOSF\_MAKE\_RESPONSE( cmd ) ((WORD)((cmd) + EKOSF\_CMD\_RESPONSE\_BASE))

#define EKOSF\_IS\_ERROR( cmd ) ((BOOL)((WORD)(cmd) >= EKOSF\_ERROR\_BASE))

// errors...

//

// generic

#define EKOSF\_ERR\_DATA\_SIZE (EKOSF\_ERROR\_BASE) // Data size in header does not equal to real data size...

#define EKOSF\_ERR\_DATA\_MISALIGN (EKOSF\_ERROR\_BASE+1) // Wrong data size alignment (not multiple of 2)...

#define EKOSF\_ERR\_DATA\_CRC (EKOSF\_ERROR\_BASE+2) // CRC data error

#define EKOSF\_ERR\_CMD\_UNKNOWN (EKOSF\_ERROR\_BASE+3) // Unknown command

#define EKOSF\_ERR\_DATA\_TOO\_LONG (EKOSF\_ERROR\_BASE+4) // pending data packet is too long

// command-related

#define EKOSF\_ERR\_DATA\_SIZE\_UNEXPECTED (EKOSF\_ERROR\_BASE+5) // actual data size is unexpected for this command

#define EKOSF\_ERR\_WRONG\_RATE (EKOSF\_ERROR\_BASE+6) // requested rate value is out of range

#define EKOSF\_ERR\_WRONG\_ADDR (EKOSF\_ERROR\_BASE+7) // requested device address is out of range

#define EKOSF\_ERR\_NOTREADY (EKOSF\_ERROR\_BASE+8) // device is not ready